Grafica 3D con Geogebra e Ti*k*Z

Luciano Battaia

Sommario

Questo articolo esamina la possibilità di utilizzare un software di geometria dinamica come Geogebra per costruire immagini geometriche tridimensionali con successiva proiezione parallela in 2D, in modo da consentire una agevole esportazione in codice PGF/Ti*k*Z.

L'utilizzo di un software come Geogebra rende possibile costruire immagini decisamente complesse senza la necessità di programmare direttamente in codice PGF/Ti*k*Z e sfruttando le potenzialità di una programmazione sostanzialmente basata sull'uso del mouse.

Sono discussi i dettagli della proiezione parallela e proposti numerosi esempi dei risultati grafici che si possono ottenere, in alcuni casi anche fornendo i codici sorgenti.

Abstract

This article considers the opportunity of using a software of dynamical geometry such as Geogebra to produce tridimensional geometric pictures with subsequent 2D parallel projection, in order to allow an easy export in PGF/Ti*k*Z code.

The use of a software such as Geogebra makes the production of very complex pictures easy enough, without any programming in PGF/Ti*k*Z code and taking advantage of a programming substantially mouse-driven.

Details of the parallel projection are discussed and numerous examples of the pictures produced are proposed, in some cases also including source codes.

1 Grafica 3D in LaTeX

La produzione di grafica vettoriale di alta qualità da inserire in un documento prodotto con LaTeX[1], in particolare annotata nello stesso stile del documento, è sempre stata una necessità essenziale per tutti gli utenti, soprattutto per la produzione di testi di carattere scientifico.

Per la grafica 2D non ci sono particolari problemi e i due strumenti più diffusi PSTricks e PGF/Ti*k*Z[2] con i loro pacchetti derivati risolvono egregiamente (quasi) ogni problema. È altresì da segnalare che molte situazioni possono essere gestite direttamente da LaTeX con l'ambiente *picture*, senza l'uso di pacchetti esterni: si veda l'articolo di Claudio Beccari (Beccari, 2011) per una estesa trattazione delle possibilità offerte da questo quasi sconosciuto ambiente.

I sorgenti contenenti codice PSTricks hanno, in LaTeX, necessariamente bisogno di una doppia compilazione per poter produrre output in pdf, ma con le tecniche oggi disponibili[3] tutto avviene in maniera automatica in background, senza bisogno di particolari interventi da parte dell'autore. I sorgenti contenenti codice Ti*k*Z, al contrario, sono direttamente compilabili in pdfLaTeX. Anche se le potenzialità teoriche di PSTricks sono superiori a quelle di Ti*k*Z, in realtà (quasi) tutto quello che si produce con PSTricks si può produrre con Ti*k*Z e la scelta tra uno dei due pacchetti è più che altro questione di gusto personale. L'autore di questo articolo ha lavorato per anni, con grande soddisfazione, solo con PSTricks[4], per poi passare progressivamente a Ti*k*Z, che ora viene usato in maniera quasi esclusiva nonostante permanga la convinzione che la scrittura di un sorgente in PSTricks sia decisamente più semplice ed intuitiva che non in Ti*k*Z. Il seguito di questo articolo conterrà esclusivamente riferimenti ed esempi in Ti*k*Z.

Una gran parte del materiale compilato con LaTeX e contenente grafica 3D utilizza software esterni di modellazione tridimensionale per produrre immagini in formato accettabile dal compilatore pdfLaTeX, immagini che poi vengono annotate con codice LaTeX extra che consente di collocare le scritte nei posti giusti: si veda a questo proposito l'articolo di Agostino de Marco (De Marco, 2008).

Altre strategie utilizzano sistemi esterni programmabili con appositi linguaggi che producono in uscita frammenti di codice anche in Ti*k*Z. L'uso di *Sketch*[5], per esempio, è discusso nell'articolo di Agostino de Marco (De Marco, 2007) e, con qualche variante peraltro molto significativa che presenteremo più avanti, è sostanzialmente una strada di questo tipo che intendiamo proporre in questo articolo. Anche TEXgraph di Patrick Fradin[6] usa questa strategia per produrre grafici molto sofisticati e complessi sia di funzioni di più variabili che di tipo più propriamente geometrico, con la possibilità di produrre frammenti di codice Ti*k*Z e, cosa molto interessante, di esportare in formato

1. In questo articolo faremo riferimento solo al processore LaTeX e non a LuaTeX, XeTeX, o altro.
2. In seguito richiamato solo con Ti*k*Z.
3. Per esempio usando il pacchetto auto-pst-pdf.
4. Si veda per esempio Battaia (2007).
5. La cui ultima versione è però del 2012.
6. http://texgraph.tuxfamily.org/

POV-Ray con cui si ottengono immagini veramente spettacolari[7] Se qualcuno ha voglia di sobbarcarsi la fatica di studiare un (complesso) linguaggio di programmazione grafica, questa potrebbe essere la soluzione giusta.

Un ulteriore approccio è quello fornito da *Asymptote*, il cui codice può essere inserito direttamente in un sorgente LaTeX mediante il pacchetto asymptote. Una ampia introduzione a questa tecnica si trova in De Marco (2009).

Esistono ancora altri approcci (*Ipe*[8] e *Inkscape*[9] per esempio) e naturalmente numerosi software commerciali: tutto questo testimonia l'importanza del problema della creazione di grafica vettoriale di alta qualità (naturalmente non solo per l'inserimento in documenti LaTeX).

Per la gestione dei grafici di funzioni di due variabili e di superfici tridimensionali si può utilizzare il pacchetto pgfplots che si appoggia a TikZ e del quale si può leggere una concisa ma efficace introduzione nell'articolo di Agostino de Marco e Roberto Giacomelli (De Marco e Giacomelli, 2011). Qui però noi siamo interessati alla creazione di grafici più propriamente "geometrici": solidi di vario tipo, in particolare poliedri, quadriche e loro intersezioni, ecc. È teoricamente anche possibile utilizzare TikZ con un po' di programmazione, come mostra l'esempio della figura 1, realizzata da Tomasz M. Trzeciak e che si può reperire su http://texample.net/tikz/examples/map-projections/.

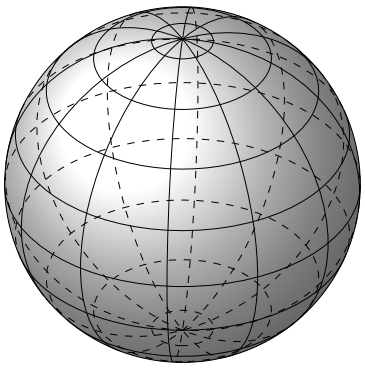

Figura 1: Sfera con meridiani e paralleli creata da Tomasz M. Trzeciak con TikZ.

Riportiamo, per un utile confronto con quanto diremo successivamente, il codice di questa figura, suddiviso in due parti: la prima da inserire nel preambolo del documento e costituita da alcuni comandi appositamente creati, la seconda da inserire nel corpo del documento e contenente le istruzioni vere e proprie per costruire la figura.

7. Esempi si trovano nel già citato sito di TEXgraph mentre un'immagine, collegata a quanto diremo e che riteniamo eccezionale, si può trovare in http://www.les-mathematiques.net/phorum/read.php?10,661628,661774.

8. http://ipe.otfried.org/

9. https://inkscape.org/it/

```
%Inserire nel preambolo del documento
\newcommand\pgfmathsinandcos[3]{
  \pgfmathsetmacro#1{sin(#3)}
  \pgfmathsetmacro#2{cos(#3)}}
\newcommand\LongitudePlane[3][current plane]{
  \pgfmathsinandcos\sinEl\cosEl{#2}
  \pgfmathsinandcos\sint\cost{#3}
  \tikzset{#1/.style={cm={\cost,%
    \sint*\sinEl,0,\cosEl,(0,0)}}}}
\newcommand\LatitudePlane[3][current plane]{
  \pgfmathsinandcos\sinEl\cosEl{#2}
  \pgfmathsinandcos\sint\cost{#3}
  \pgfmathsetmacro\yshift{\cosEl*\sint}
  \tikzset{#1/.style={cm={\cost,0,0,%
    \cost*\sinEl,(0,\yshift)}}}}
\newcommand\DrawLongitudeCircle[2][1]{
  \LongitudePlane{\angEl}{#2}
  \tikzset{current plane/.prefix %
    style={scale=#1}}
  \pgfmathsetmacro\angVis{atan(sin(#2)*%
    cos(\angEl)/sin(\angEl))}
  \draw[current plane] (\angVis:1) %
    arc (\angVis:\angVis+180:1);
  \draw[current plane,dashed](\angVis-180:1)%
    arc (\angVis-180:\angVis:1);}
\newcommand\DrawLatitudeCircle[2][1]{
  \LatitudePlane{\angEl}{#2}
  \tikzset{current plane/.prefix %
    style={scale=#1}}
  \pgfmathsetmacro\sinVis{sin(#2)/cos(#2)*%
    sin(\angEl)/cos(\angEl)}
  \pgfmathsetmacro\angVis{asin(min(1,%
    max(\sinVis,-1)))}
  \draw[current plane] (\angVis:1) %
    arc (\angVis:-\angVis-180:1);
  \draw[current plane,dashed](180-\angVis:1)%
    arc (180-\angVis:\angVis:1);}

%Inserire nel corpo del documento
\begin{tikzpicture}
\def\R{2.5}
\def\angEl{35}
\filldraw[ball color=white](0,0)circle(\R);
\foreach \t in {-80,-60,...,80} &
  { \DrawLatitudeCircle[\R]{\t} }
\foreach \t in {-5,-35,...,-175} %
{ \DrawLongitudeCircle[\R]{\t} }
\end{tikzpicture}
```

È proprio l'esame del codice di questa figura che ci ha stimolato ad approfondire lo studio di TikZ per verificare la possibilità di creare le figure di cui avevamo bisogno. Purtroppo il manuale di TikZ di Till Tantau, è estremamente voluminoso e la curva di apprendimento, dopo una pendenza iniziale abbastanza lieve, si impenna rapidamente. Abbiamo perciò deciso di tentare altre strade.

2 L'intervento di Geogebra

Per motivi didattici abbiamo da sempre estesamente usato diversi software di geometria dinamica, principalmente *Cabri* e, successivamente, *Geogebra*, soprattutto per il fatto che il

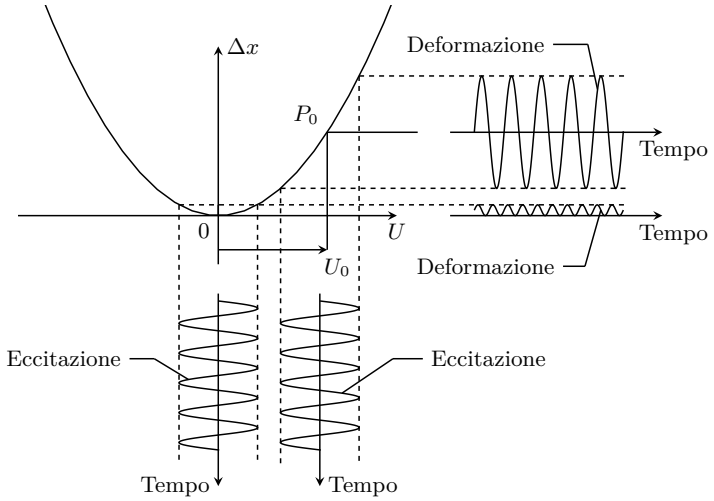

FIGURA 2: Una figura prodotta con Geogebra ed esportata in TikZ, realizzata per una tesi di laurea in fisica.

suo uso è gratuito per scopi non commerciali. L'utilizzo basico di questo software è talmente semplice che può essere adatto anche a bambini di scuola elementare: si veda per esempio lo splendido sito *Splashragazzi*, all'indirizzo `http://splashscuola.altervista.org/esercizi/geogebra/geogebra_elementare.shtml`.

La cosa interessante per quanto ci riguarda è la possibilità di costruire figure complesse bidimensionali ed esportarle producendo codici[10] TikZ perfettamente funzionanti, anzi se non molto "puliti": alcuni semplici adattamenti, in particolare riguardo la posizione delle etichette, possono soddisfare comunque molte esigenze dei produttori di documenti contenenti grafica 2D di alta qualità, senza conoscere quasi per nulla i complicati meandri della programmazione diretta in TikZ. Il vantaggio nell'uso di Geogebra è principalmente legato al fatto che la maggior parte delle costruzioni può avvenire con il mouse, senza ricorrere all'uso di alcun codice. Solamente per richieste abbastanza complesse è necessaria l'immissione di alcune istruzioni in una apposita riga di comando. Anche se la filosofia di questa produzione è piuttosto del tipo WYSIWYG, quindi a priori lontana da quella di un utente LaTeX puro, riteniamo che per la produzione di immagini sia decisamente da preferire. La quasi totalità delle immagini bidimensionali che abbiamo prodotto negli anni per i testi di appunti di argomento matematico per il triennio terminale del Liceo Scientifico e i primi anni di università (corsi di laurea in Ingegneria, Economia, Tecnologie multimediali) sono stati prodotti con questo sistema, e si tratta di migliaia di figure. L'avvento di questa possibilità ha reso per noi obsoleto l'uso di pacchetti per altro molto ben fatti come per esempio tkz-euclide, o il suo predecessore della serie PSTricks, pst-eucl.

Con un po' di conoscenze più approfondite di Geogebra si possono realizzare con pochi click del mouse e alcuni comandi molto semplici anche figure molto complesse. A titolo d'esempio si veda la figura 2, realizzata per una tesi di laurea specialistica in fisica.

Da qualche tempo è uscita la versione 5 di Geogebra che consente anche la costruzione di figure tridimensionali e la trattazione di funzioni in due variabili e di superfici dello spazio. Purtroppo per la versione 3D non è prevista l'esportazione in codice TikZ (e nemmeno in altri formati) e non crediamo che la cosa sarà possibile nemmeno in futuro, perlomeno in un futuro non troppo lontano.

A questo punto è venuto spontaneo chiedersi: se non è possibile l'esportazione diretta delle figure 3D, perché non realizzare direttamente in Geogebra una proiezione di qualche tipo da 3D a 2D e poi esportare quest'ultima? In fondo ogni figura 3D non è altro che una opportuna proiezione 2D di un oggetto tridimensionale.

Nella successiva ricerca in internet per valutare l'opportunità e la fattibilità di una idea del genere ci siamo imbattuti nell'interessante articolo di Keith Wolcott (WOLCOTT, 2012), in cui sono discussi numerosi problemi della grafica 3D con TikZ, con la proposta di alcune soluzioni. Il lavoro di Wolcott prende le mosse dalla necessità di costruire un grafico che evidenzi la circonferenza intersezione di due sfere. Viene estesamente usato il codice di Tomasz M. Trzeciak, di cui abbiamo già parlato, per creare l'immagine di una sfera con meridiani e paralleli e l'articolo si conclude con l'immagine desiderata, che riproduciamo nella figura 3.

Purtroppo, come lo stesso Wolcott riconosce, la figura non è corretta, in quanto non sono gestite accuratamente le parti delle due sfere che si sovrap-

10. Geogebra è anche in grado di produrre codici PSTricks e *Asymptote*, ma non ci occupiamo qui di questa capacità.

pongono. Questo fatto rende evidente le difficoltà della gestione di questo tipo di problemi con il solo codice TikZ (gestione che riteniamo comunque possibile), e dunque la necessità di cercare altre soluzioni.

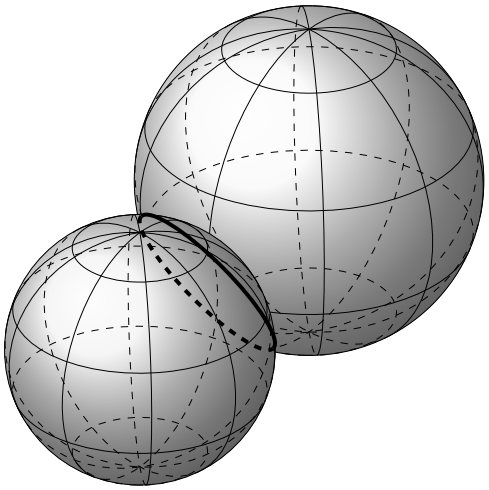

Figura 3: Intersezione di due sfere in TikZ, come presentata nell'articolo di Keith Wolcott.

La prima cosa che abbiamo cercato di realizzare è la riproduzione della sfera di Tomasz M. Trzeciak, utilizzando Geogebra e limitando all'essenziale la programmazione in codice TikZ. Il risultato ottenuto è proposto nella figura 4: in seguito saranno proposti i dettagli su come ottenerla.

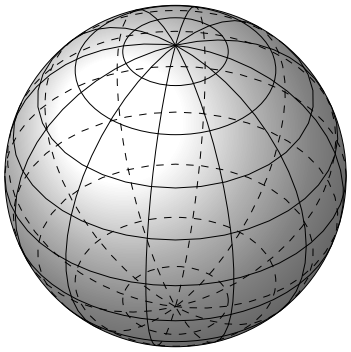

Figura 4: Sfera con meridiani e paralleli, costruita esportando il codice da Geogebra.

Anche di questa figura proponiamo il codice sorgente, nella figura 6. Nel seguito non proporremo altri codici: la cosa sarebbe poco significativa, in quanto essi vengono prodotti in automatico da Geogebra.

L'intervento manuale sul codice prodotto da Geogebra è limitato all'introduzione della riga contenente il comando `\shadedraw`, che serve a produrre l'ombreggiatura. È immediato constatare che il codice di Trzeciak, sopra riportato, è decisamente più elegante e conciso, ma non bisogna dimenticare che il codice di Geogebra è, appunto, prodotto in automatico da Geogebra, quasi senza interventi da parte dell'utente. Questo codice è meno pulito anche perché usa solo comandi elementari di TikZ, in particolare quasi esclusivamente il comando `\draw`. C'è anche un'altra osservazione da fare: in un documento in cui ci sono decine di figure il preambolo del documento, dove vanno inseriti i nuovi comandi, diventa decisamente complesso (rendendo quindi conveniente avere un file separato che li raggruppi) e la gestione di eventuali modifiche potrebbe diventare problematica.

Dal punto di vista del risultato finale ci pare che non sia percepibile alcuna differenza tra le sfere delle figure 1 e 4.

Prima di passare ai dettagli della tecnica che intendiamo proporre è doveroso citare il pacchetto tikz-3dplot di Jeff Hein che in sostanza implementa alcune macro in linguaggio TikZ al fine di disegnare, in maniera relativamente semplice, sistemi di coordinate tridimensionali e semplici figure tridimensionali. Il pacchetto è stato sviluppato dall'autore per la sua esigenza di produrre accurate immagini tridimensionali di vettori e, in questo campo, raggiunge pienamente lo scopo. Il suo utilizzo per situazioni più generali, pur se in teoria possibile, è nella pratica decisamente complesso. Riportiamo, nella figura 5, un esempio d'uso prodotto da uno studente di fisica e il cui codice si può reperire in `http://tex.stackexchange.com/questions/39577/`.

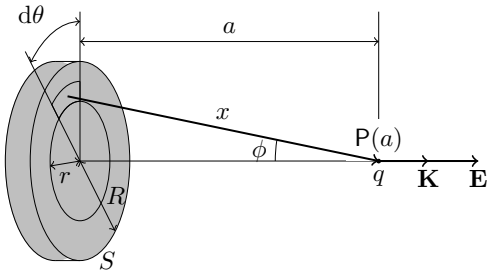

Figura 5: Una figura 3D prodotta con il pacchetto tikz-3dplot.

Anche con questo pacchetto, tuttavia, vi è la necessità di programmazione (anche abbastanza complessa) in codice TikZ e, inoltre, le figure a cui siamo interessati sono ancora difficilmente trattabili.

3 Un po' di matematica dietro le quinte

Geogebra è un software di geometria dinamica[11] molto articolato e con numerose possibilità. In so-

11. Riportiamo esplicitamente la Licenza d'uso: "You are free to copy, distribute and transmit GeoGebra free of charge for non-commercial purposes". Dettagli e condizioni si trovano nella finestra di guida del programma.

```
\begin{tikzpicture}[x=0.8cm,y=0.8cm]
\clip(-3.,-1.5) rectangle (5.,5.5);
\shadedraw[ball color = white](1.,2.) circle (3.);
\draw [shift={(1.,2.)}] plot[domain=-3.142:0.,variable=\t]({-0.87*3.*cos(\t r)+%
0.493*2.16*sin(\t r)},{-0.493*3.*cos(\t r)+-0.87*2.16*sin(\t r)});
\draw [shift={(1.,2.)},dash pattern=on 3pt off 3pt]  plot[domain=0.:3.142,variable=\t]%
({-0.87*3.*cos(\t r)+0.493*2.16*sin(\t r)},{-0.493*3.*cos(\t r)+-0.87*2.16*sin(\t r)});
\draw [shift={(1.,2.)},dash pattern=on 3pt off 3pt]  plot[domain=0.:3.142,variable=\t]%
({-0.475*3.*cos(\t r)+0.88*1.4772116295183126*sin(\t r)},{-0.88*3.*cos(\t r)+-%
0.475*1.477*sin(\t r)});
\draw [shift={(1.,2.)}] plot[domain=-3.142:0.,variable=\t]({-0.475*3.*cos(\t r)+%
0.88*1.477*sin(\t r)},{-0.88*3.*cos(\t r)+-0.475*1.477*sin(\t r)});
\draw [shift={(1.,2.)},dash pattern=on 3pt off 3pt]  plot[domain=0.:3.142,variable=\t]%
({-0.113*3.*cos(\t r)+0.994*0.3990666646784674*sin(\t r)},{-0.994*3.*cos(\t r)+-%
0.113*0.399*sin(\t r)});
\draw [shift={(1.,2.)}]  plot[domain=-3.142:0.,variable=\t]({1.*3.*cos(\t r)+%
0.*1.928*sin(\t r)},{0.*3.*cos(\t r)+1.*1.928*sin(\t r)});
\draw [shift={(1.,2.)}] plot[domain=-3.142:0.,variable=\t]({-0.113*3.*cos(\t r)+%
0.994*0.399*sin(\t r)},{-0.994*3.*cos(\t r)+-0.113*0.399*sin(\t r)});
\draw [shift={(1.,2.)},dash pattern=on 3pt off 3pt]  plot[domain=0.:3.142,variable=\t]%
({-0.228*3.*cos(\t r)+-0.974*0.786*sin(\t r)},{0.974*3.*cos(\t r)+-0.228*0.786*sin(\t r)});
\draw [shift={(1.,2.)}] plot[domain=-3.142:0.,variable=\t]({-0.228*3.*cos(\t r)+-%
0.974*0.786*sin(\t r)},{0.974*3.*cos(\t r)+-0.228*0.786*sin(\t r)});
\draw [shift={(1.,2.)},dash pattern=on 3pt off 3pt]  plot[domain=0.:3.142,variable=\t]%
({-0.608*3.*cos(\t r)+-0.794*1.76*sin(\t r)},{0.794*3.*cos(\t r)+-0.608*1.76*sin(\t r)});
\draw [shift={(1.,2.)}] plot[domain=-3.142:0.,variable=\t]({-0.608*3.*cos(\t r)+-%
0.794*1.76*sin(\t r)},{0.794*3.*cos(\t r)+-0.608*1.76*sin(\t r)});
\draw [shift={(1.,2.)},dash pattern=on 3pt off 3pt]  plot[domain=0.:3.142,variable=\t]%
({-0.964*3.*cos(\t r)+-0.265*2.263*sin(\t r)},{0.265*3.*cos(\t r)+-0.964*2.263*sin(\t r)});
\draw [shift={(1.,2.)}] plot[domain=-3.142:0.,variable=\t]({-0.964*3.*cos(\t r)+-%
0.265*2.263*sin(\t r)},{0.265*3.*cos(\t r)+-0.964*2.263*sin(\t r)});
\draw [rotate around={0.:(1.,4.186)}] (1.,4.186) ellipse (0.927 and 0.596);
\draw [rotate around={0.:(1.,3.86)}] (1.,3.86) ellipse (1.763 and 1.133);
\draw [rotate around={0.:(1.,0.141)},dash pattern=on 3pt off 3pt] (1.,0.141) %
ellipse (1.763 and 1.133);
\draw [rotate around={0.:(1.,-0.186)},dash pattern=on 3pt off 3pt] (1.,-0.186) &
ellipse (0.927 and 0.596);
\draw [shift={(1.,3.35)}] plot[domain=-3.797:0.656,variable=\t]({1.*2.427*cos(\t r)+%
0.*1.56*sin(\t r)},{0.*2.427*cos(\t r)+1.*1.56*sin(\t r)});
\draw [shift={(1.,3.35)},dash pattern=on 3pt off 3pt]  plot[domain=0.656:2.486,variable=%
\t]({1.*2.427*cos(\t r)+0.*1.56*sin(\t r)},{0.*2.427*cos(\t r)+1.*1.56*sin(\t r)});
\draw [shift={(1.,2.71)}] plot[domain=-3.418:0.276,variable=\t]({1.*2.853*cos(\t r)+%
0.*1.834*sin(\t r)},{0.*2.853*cos(\t r)+1.*1.834*sin(\t r)});
\draw [shift={(1.,2.71)},dash pattern=on 3pt off 3pt]  plot[domain=0.276:2.865,variable=%
\t]({1.*2.853*cos(\t r)+0.*1.834*sin(\t r)},{0.*2.853*cos(\t r)+1.*1.834*sin(\t r)});
\draw [shift={(1.,2.)},dash pattern=on 3pt off 3pt]  plot[domain=0.:3.142,variable=\t]%
({1.*3.*cos(\t r)+0.*1.928*sin(\t r)},{0.*3.*cos(\t r)+1.*1.928*sin(\t r)});
\draw [shift={(1.,1.29)}] plot[domain=3.418:6.007,variable=\t]({1.*2.853*cos(\t r)+%
0.*1.834*sin(\t r)},{0.*2.853*cos(\t r)+1.*1.834*sin(\t r)});
\draw [shift={(1.,1.29)},dash pattern=on 3pt off 3pt]  plot[domain=-0.276:3.418,variable=%
\t]({1.*2.853*cos(\t r)+0.*1.834*sin(\t r)},{0.*2.853*cos(\t r)+1.*1.834*sin(\t r)});
\draw [shift={(1.,0.649)}] plot[domain=3.797:5.628,variable=\t]({1.*2.427*cos(\t r)+%
0.*1.56*sin(\t r)},{0.*2.427*cos(\t r)+1.*1.56*sin(\t r)});
\draw [shift={(1.,0.649)},dash pattern=on 3pt off 3pt]  plot[domain=-0.656:3.797,variable=%
\t]({1.*2.427*cos(\t r)+0.*1.56*sin(\t r)},{0.*2.427*cos(\t r)+1.*1.56*sin(\t r)});
\end{tikzpicture}
```

FIGURA 6: Listato del codice per la sfera della figura 4, codice quasi integralmente generato da Geogebra.

stanza esso prevede[12] due finestre per grafici 2D, una finestra per grafici 3D, una finestra contenente un foglio di calcolo abbastanza potente, una finestra CAS per calcoli numerici e algebrici (anche simbolici), un calcolatore di probabilità, una finestra di algebra dove sono proposte le coordinate dei punti, le equazioni delle curve, le lunghezze dei

12. Almeno nella versione 5.0.268 attualmente disponibile: il software è in rapido sviluppo e non sono escluse novità anche nel breve periodo.

segmenti, ecc., rappresentati nelle finestre grafiche. Tutte le finestre possono interagire tra di loro: per quanto ci interessa, per esempio, tutto quanto ottenuto nella finestra 3D può essere opportunamente trasferito alla finestra 2D. Le opzioni non finiscono qui: c'è, per esempio, la possibilità di una programmazione Javascript, ma non intendiamo occuparci qui in dettaglio di questo. Infine è prevista una riga di comando per tutto quanto non si può definire direttamente con il mouse. Segnaliamo infine che eventuali annotazioni testuali nelle varie finestre grafiche si inseriscono con codice LaTeX.

L'idea di base per le costruzioni che vogliamo realizzare è, come già più sopra accennato, di eseguire una proiezione parallela da 3D a 2D direttamente in Geogebra per poi esportarla, usando l'apposita funzione di Geogebra, in codice TikZ. Supposto dunque di avere un sistema cartesiano ortogonale nello spazio tridimensionale (che Geogebra visualizza nella finestra 3D) con l'asse z verticale ascendente, indichiamo con α una rotazione attorno all'asse verticale e con β una rotazione attorno ad un asse orizzontale. Si può allora costruire nel piano 2D il triedro proiezione, i cui vettori di base indichiamo con $\vec{i}$, $\vec{j}$, $\vec{k}$, con le seguenti formule:

$$\begin{aligned} \vec{i} &= \big(-\cos(\alpha), -\sin(\alpha)\sin(\beta)\big) \\ \vec{j} &= \big(\sin(\alpha), -\cos(\alpha)\sin(\beta)\big) \\ \vec{k} &= \big(0, \cos(\beta)\big) \end{aligned} \quad .$$

Queste formule possono essere scritte in diverse (limitate) varianti: abbiamo scelto quella che ci era più famigliare in quanto è quella usata da Herbert Voss (Voss, 2014) per il suo pacchetto (molto interessante) pst-3dplot, pacchetto che abbiamo estesamente usato prima di scoprire le possibilità offerte da Geogebra.

Se si fissa l'origine in $\mathsf{O} = (0, 0)$, come conviene, in Geogebra si devono creare due "slider" angolari con i nomi α e β, dopodiché i vettori indicati si costruiscono con i seguenti comandi, dove abbiamo tenuto conto del fatto che Geogebra non usa particolari formattazioni per la scrittura dei vettori,

$$\begin{aligned} \mathrm{i} &= \text{Vettore}[\mathsf{O}, (-\cos(\alpha), -\sin(\alpha)\sin(\beta))] \\ \mathrm{j} &= \text{Vettore}[\mathsf{O}, (\sin(\alpha), -\cos(\alpha)\sin(\beta))] \\ \mathrm{k} &= \text{Vettore}[\mathsf{O}, (0, \cos(\beta))] \end{aligned}$$

Fatto questo, se si ha un punto $\mathsf{P} = (x_\mathsf{P}, y_\mathsf{P}, z_\mathsf{P})$ nella finestra 3D, il suo corrispondente nella proiezione sarà

$$\mathsf{P}' = x_\mathsf{P}\,\vec{i} + y_\mathsf{P}\,\vec{j} + z_\mathsf{P}\,\vec{k},$$

ovvero, nel linguaggio di Geogebra,

$$\mathsf{P}' = x(\mathsf{P})\,\mathrm{i} + y(\mathsf{P})\,\mathrm{j} + z(\mathsf{P})\,\mathrm{k}.$$

Se invece si ha una curva C di equazioni parametriche[13] $(f(t), g(t), h(t))$, con il parametro t opportunamente variabile tra due estremi, la sua proiezione 2D avrà, sempre nel linguaggio di Geogebra, equazioni parametriche

$$\begin{aligned} x' &= f(t)\,x(\mathrm{i}) + g(t)\,x(\mathrm{j}) + h(t)\,x(\mathrm{k}) \\ y' &= f(t)\,y(\mathrm{i}) + g(t)\,y(\mathrm{j}) + h(t)\,y(\mathrm{k}) \end{aligned} \quad .$$

In sostanza usando queste formule si può costruire, come vedremo in dettaglio su qualche esempio, la proiezione 2D di un qualunque grafico realizzato in Geogebra 3D.

A questo punto, lavorando nella finestra 2D di Geogebra, si può scegliere la migliore vista per la figura in esame, variando opportunamente gli angoli α e β: è anche questa una delle opportunità offerte dall'uso di Geogebra, in quanto è molto difficile in generale decidere a priori qual è l'angolo di visuale più opportuno per una data immagine, e solo sperimentando dinamicamente si riesce a trovare la soluzione adatta.

Naturalmente nemmeno Geogebra banalizza il problema della grafica 3D. In effetti, è chiaro che chi ha bisogno di immagini di questo tipo deve avere una adeguata preparazione matematica, in particolare deve poter maneggiare con disinvoltura vettori ed equazioni di curve e superfici dello spazio: nulla si ottiene gratuitamente!

Alla luce di queste formule vediamo come si può ottenere l'immagine della sfera proposta nella figura 4. Costruita, nella finestra 3D, la sfera di centro l'origine e raggio r, se ne costruisce la proiezione 2D, che è semplicemente la circonferenza di centro l'origine e raggio r. Si procede poi con i paralleli e i meridiani. Supponiamo di voler costruire 5 paralleli. Essi si troveranno alle seguenti latitudini: $-60°$, $-30°$, $0°$, $30°$, $60°$. I piani che li contengono avranno equazioni, rispettivamente,

$$\begin{aligned} z &= r\sin(-60°) \;;\; z = -r\sqrt{3}/2 \\ z &= r\sin(-30°) \;;\; z = -r/2 \\ z &= r\sin(0°) \;;\; z = 0 \\ z &= r\sin(30°) \;;\; z = r/2 \\ z &= r\sin(60°) \;;\; z = r\sqrt{3}/2 \end{aligned} \quad ,$$

piani che si possono tracciare nella finestra 3D semplicemente scrivendone l'equazione nella barra di inserimento.

Per ciascuno di essi si fa costruire a Geogebra l'intersezione sfera-piano e, nella circonferenza di intersezione, si scelgono con il mouse 5 punti a caso; determinate, con le formule sopra riportate, le proiezioni di questi punti nella finestra 2D, si costruisce, con l'apposito comando di Geogebra, la conica per essi: si avrà la proiezione 2D del parallelo. Analogo discorso per i meridiani.

13. Geogebra è in grado di gestire curve in equazione parametrica anche molto complesse.

Una volta tracciati tutti i meridiani e i paralleli voluti si sceglie dinamicamente qual è l'angolo di visuale desiderato. Successivamente, per ciascuno di essi si stabilisce qual è la parte visibile e quale quella invisibile scegliendo, per quest'ultima, se visualizzarla in tratteggio e magari con uno spessore di linea ridotto, o non visualizzarla affatto. A questo punto basta solo esportare il tutto con l'apposito comando di Geogebra in formato TikZ, scegliendo quale parte della finestra esportare: il codice prodotto è direttamente funzionante e vi si possono apportare le modifiche desiderate. Generalmente le eventuali etichette non sono piazzate nella maniera migliore, ma è molto facile adeguare il piazzamento ai propri gusti.

In realtà con un po' di esperienza nell'uso di Geogebra il processo che abbiamo descritto può essere automatizzato in gran parte. Per esempio è possibile costruire in modo ricorsivo i meridiani e i paralleli, utilizzando il comando Successione di Geogebra, ma abbiamo voluto qui proporre una tecnica assolutamente di base.

4 Intersezione di due sfere

Ritorniamo all'immagine dell'intersezione di due sfere proposta da Keith Wolcott.

In questo caso si tratta di disegnare due sfere con i loro meridiani e paralleli come sopra descritto, tenendo conto che una delle due sfere può avere centro l'origine, l'altra è meglio che abbia un centro variabile parametricamente[14] in modo da poter poi scegliere la visuale più adatta ad evidenziare la circonferenza intersezione. È opportuno che anche il raggio delle sfere (o almeno di una della due) sia parametrizzato, per la stessa ragione di scelta della visuale.

Fatta disegnare da Geogebra la circonferenza intersezione, si scelgono su di essa cinque punti e, con la tecnica già descritta sopra, si trova la sua ellisse immagine in 2D: a questo punto non resta che scegliere la visuale più opportuna e, successivamente, per ciascun meridiano, parallelo e per l'ellisse intersezione la parte visibile e quella invisibile, decidendo come visualizzarla. Anche se quest'ultima parte è un po' noiosa (e richiede una pazienza certosina!), non ha bisogno di alcuna tecnica particolare: si tratta di un'operazione di base in Geogebra.

Il risultato che abbiamo ottenuto è mostrato nella figura 7.

La figura 7 mostra la corretta gestione delle parti di sfera sovrapposte e, a nostro modo di vedere, presenta un angolo di visuale particolarmente adatto al caso. Abbiamo scelto di non disegnare affatto, nemmeno con un tratteggio speciale o con una linea sottile, le parti di sfera sovrapposte, e questo per una migliore leggibilità della figura. Non costerebbe alcuna fatica ulteriore visualizzare anche qualcosa di queste parti nascoste, particolarmente se si volesse produrre una figura di dimensioni molto più grandi, ed è anche questa, ancora una volta, una delle opportunità offerte facilmente dal tipo di approccio che stiamo proponendo.

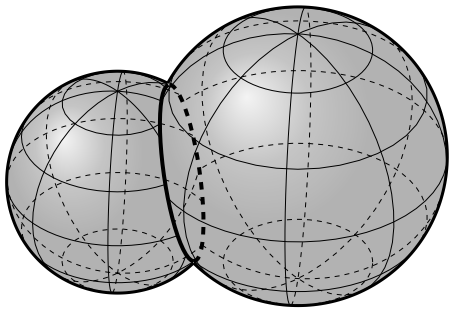

Figura 7: Intersezione di due sfere ottenuta mediante codice esportato da Geogebra.

5 Spirale su una sfera

L'esempio che proponiamo ora richiede un minimo di matematica in più, ma nessuna ulteriore fatica dal punto di vista della costruzione del codice. Si tratta di disegnare una linea complessa, in questo caso una spirale con infinite spire, su una sfera. In situazioni come queste è indispensabile utilizzare le equazioni parametriche della curva in questione, equazioni che si possono comunque trovare facilmente in rete, per esempio sul famoso *MathWorld* di Wolfram[15]. Se la sfera ha centro l'origine, raggio r e a è una costante, le equazioni sono le seguenti

$$\begin{aligned} x(t) &= \frac{\cos t}{\sqrt{1+a^2t^2}} \\ y(t) &= \frac{\sin t}{\sqrt{1+a^2t^2}} \\ z(t) &= \frac{-at}{\sqrt{1+a^2t^2}} \end{aligned} .$$

Nell'immagine che proponiamo abbiamo scelto $r = 1.8$ e $a = 0.12$. Al solito se queste quantità sono impostate parametricamente, si può sperimentare con l'immagine in Geogebra al fine di scegliere i valori più opportuni e solo alla fine esportare il risultato. Il tracciamento della curva, sia in 3D (cosa di poco interesse in questo caso) sia in 2D non richiede alcuna particolare tecnica: basta usare le formule per la proiezione 2D di una curva parametrica 3D, formule che abbiamo già introdotto. Al solito la cosa che resta da fare è valutare quali parti sono visibili e quali sono invisibili, e come renderle nell'immagine da inserire nel documento LaTeX.

Il risultato è mostrato nella figura 8.

14. Operazione di routine in Geogebra!

15. http://mathworld.wolfram.com/

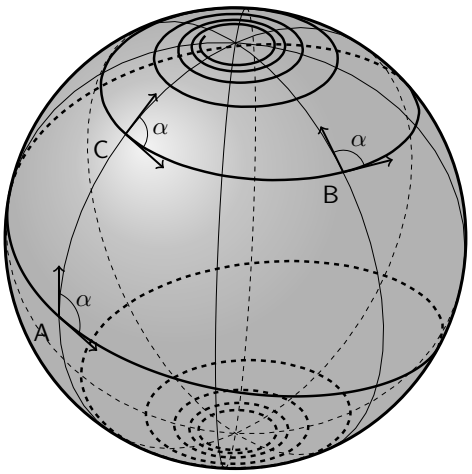

Figura 8: Spirale tracciata su una sfera. Codice ottenuto per esportazione da Geogebra.

Se il parametro t nelle equazioni della spirale tende all'infinito, la spirale stessa tende al polo Sud e al polo Nord della sfera.

È d'obbligo una ulteriore precisazione su questa figura, come su analoghe che coinvolgono il tracciamento di curve complesse, ma questo è un problema collegato alla struttura di TikZ[16], e non ha niente a che fare con le difficoltà di programmazione nel linguaggio del pacchetto. Il problema è che curve complesse non sono tracciabili con i comandi di TikZ e bisogna ricorrere a programmi esterni che generino una matrice di punti per il tracciamento. La scelta più semplice è quella di usare GNUPLOT. Per fare questo occorre avere installato il programma, che è distribuito con una sua propria licenza di software free. Occorre inoltre avere abilitato il compilatore scelto per LaTeX all'uso di programmi esterni, per esempio con pdflatex –shell-escape <nomefile>.tex. Tuttavia, a parte l'abilitazione del compilatore che si deve fare una volta per tutte e che serve anche per altre cose, Geogebra gestisce automaticamente nell'esportazione in TikZ anche questo fatto!

La figura 8 evidenzia la proprietà fondamentale di questo tipo di spirale, e cioè il fatto che in ogni punto l'angolo tra la spirale stessa e il meridiano passante per quel punto è costante. Si noti anche che dalla stessa figura si evince che la proiezione parallela non mantiene gli angoli, come è ovvio, ma mantiene il parallelismo, come si può notare se si osservano nei due punti A e C due dei quattro vettori che sono paralleli nello spazio e che rimangono paralleli nella proiezione.

16. Anzi, ancora più correttamente, allo stesso motore matematico di LaTeX.

6 Poliedri

Nell'esperienza di produzione di grafica 3D che abbiamo fin qui accumulato, uno dei campi in cui l'intervento di Geogebra è stato davvero provvidenziale è quello dei poliedri e dei loro sviluppi.

Geogebra implementa infatti delle routine per la costruzione di poliedri e in particolare dei solidi platonici con la visualizzazione dinamica dei loro sviluppi. Una volta costruito un poliedro regolare e il suo sviluppo (ad un qualunque stadio) nella finestra 3D, è sufficiente proiettare i vertici uno ad uno nel piano 2D e poi congiungerli con segmenti (operazione standard in Geogebra) per ottenere la figura 2D richiesta. Non rimane, come già più volte ricordato, che scegliere il miglior angolo di visuale possibile, valutare quali sono le parti visibili e quali quelle invisibili, decidendo come rendere queste ultime ed esportare il tutto in formato TikZ. Qui bisogna poi intervenire sul codice per inserire le ombreggiature, tenendo conto del punto da cui si vuol fare provenire la luce.

Proponiamo nella figura 9 l'immagine del dodecaedro ottenuta con la tecnica descritta. Non ci si lasci ingannare dall'apparente semplicità della figura: un calcolo "manuale" della posizione[17] dei vertici del dodecaedro non è per niente semplice, meno che mai il relativo calcolo se fatto eseguire direttamente dal codice TikZ.

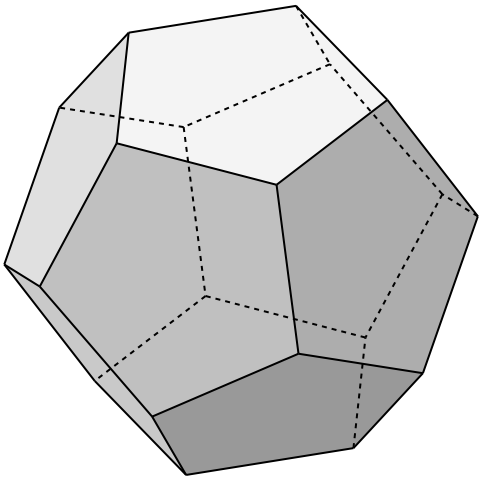

Figura 9: Il dodecaedro regolare. Figura ottenuta mediante esportazione di codice da Geogebra.

La figura 10 mostra invece un primo passo nello sviluppo dello stesso dodecaedro. In questo caso la determinazione manuale o mediante codice TikZ della posizione dei vari vertici è ovviamente ancora più complessa (anche se naturalmente possibile).

17. Per chi avesse voglia di cimentarsi con questo problema suggeriamo la bellissima costruzione che Euclide propone nel Libro XIII dei suoi Elementi, alla proposizione 17.

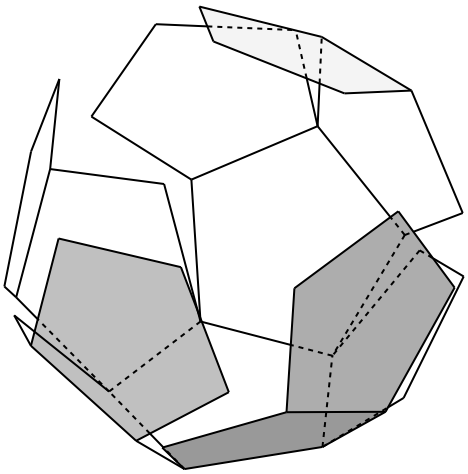

FIGURA 10: Il dodecaedro regolare: un primo passo nello sviluppo piano.

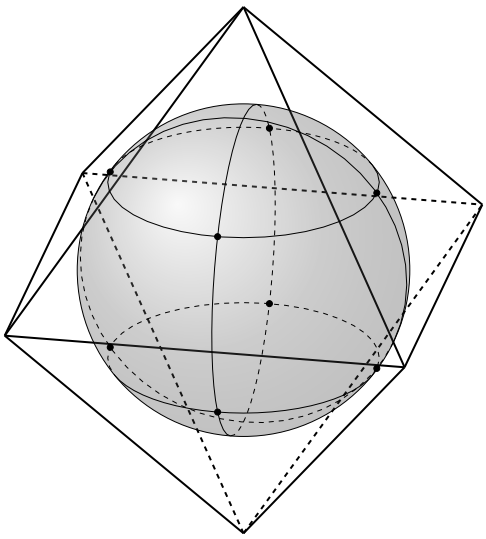

FIGURA 11: L'ottaedro regolare e la sfera inscritta. Sono evidenziati i meridiani e i paralleli passanti per quattro degli otto punti di tangenza, nonché i punti di tangenza stessi.

Con la tecnica basata su Geogebra, invece, non si introduce praticamente nessuna difficoltà ulteriore rispetto alla costruzione della figura originale. Anzi, se si programma correttamente la figura in Geogebra, la figura 10 si ottiene dalla 9 in pochi minuti.

Non ci sono particolari difficoltà nemmeno nel rappresentare la sfera inscritta o circoscritta a uno dei poliedri regolari: nella figura 11 proponiamo, a titolo d'esempio, il caso dell'ottaedro e della sfera inscritta. Si deve pensare che l'ottaedro sia trasparente per la sfera e i suoi meridiani e paralleli, mentre l'ottaedro e la sfera sono opachi per i loro lati o parti di meridiani e paralleli invisibili: è questa una scelta puramente personale e che può essere comunque facilmente modificata.

Nella figura 12 proponiamo la trattazione dell'ottaedro in una vista diversa da quella della figura 11 e di uno dei suoi sviluppi piani. Tra i tanti possibili sviluppi ne abbiamo scelto uno classico, ovvero quello proposto originariamente da Daniel Barbaro (BARBARO, 1769) nel Capitolo IIII, Parte Terza, pp. 48-49, col titolo "SPIEGATURA, DRITTO, ET ADOMBRATIONE del corpo detto octoedro".

La cosa interessante è che, sempre sfruttando le capacità di Geogebra, è facile anche costruire le curve descritte dai vertici del poliedro durante lo sviluppo.

Infatti, nello sviluppo di un poliedro in Geogebra è possibile introdurre un parametro variando il quale si possono vedere dinamicamente le varie fasi dello sviluppo stesso. Si può impostare Geogebra in modo da far lasciare una traccia ai vertici interessati durante il moto ed è anche possibile visualizzare questa traccia nella proiezione 2D: a questo punto è facile costruire, con una macro di Geogebra, una cubica di Bézier che approssimi, magari a tratti, la traiettoria descritta. Questa curva è direttamente esportabile in linguaggio TikZ, ottenendo così la sua rappresentazione nella figura LaTeX.

Quasi sempre sarebbe anche possibile ricavare le equazioni parametriche delle traiettorie descritte dai vertici nello sviluppo, tuttavia ciò richiederebbe conoscenze non elementari di meccanica dei corpi rigidi: per esempio, con riferimento alla figura 12, la curva descritta dal vertice A risulta dalla composizione di un moto di rotazione del triangolo ABC attorno allo spigolo BC, e da un moto di rotazione del triangolo OBC attorno al vertice fisso O. La tecnica descritta che usa le tracce lasciate dal vertice A durante il suo moto è invece elementare e non richiede alcun supplemento di calcolo, permettendo di ottenere facilmente la curva Γ.

7 Il pallone da calcio

Una volta acquisita dimestichezza con i solidi platonici si può esperimentare l'ampliamento della tecnica al caso di altri solidi, per esempio i cosiddetti solidi archimedei che si differenziano dai solidi platonici sostanzialmente[18] perché le facce, pur essendo ancora costituite tutte da poligoni regolari, non sono tutte uguali, ma sono costituite da due o tre tipi di poligoni regolari. Tuttavia essi mantengono la proprietà dei solidi platonici di avere gli spigoli tutti uguali.

18. In realtà la definizione di solido archimedeo è un po' più complessa, ma non intendiamo qui approfondire la questione. Del resto anche la nozione di poliedro regolare è tutt'altro che banale: per esempio, unendo per una faccia due tetraedri regolari si ottiene un poliedro che ha sei facce uguali e tutte costituite da poligoni (triangoli equilateri) regolari, ma non è un poliedro regolare.

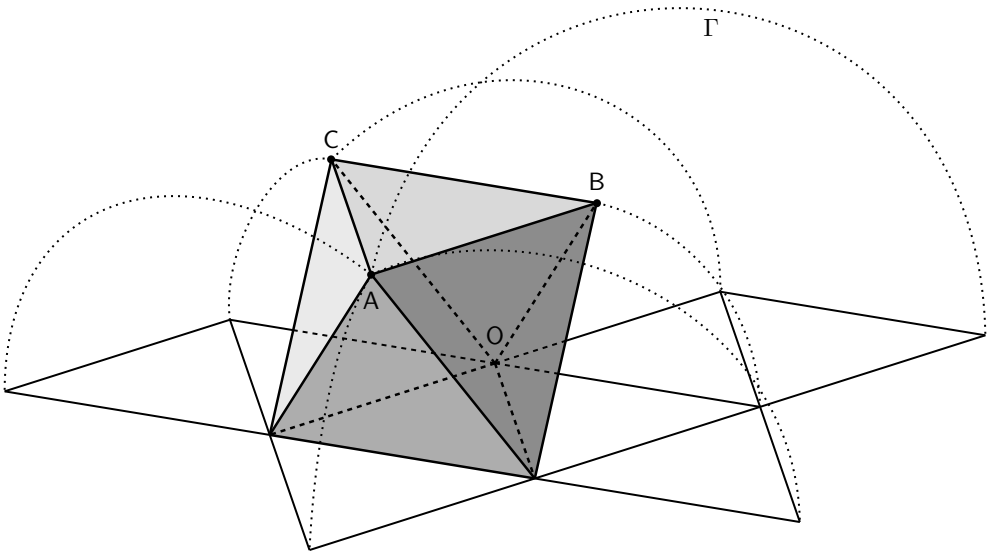

Figura 12: Uno dei possibili sviluppi piani dell'ottaedro: lo sviluppo completo in 3D, con evidenziazione delle curve descritte dai vertici nello sviluppo.

La tecnica più semplice per costruire alcuni di questi poliedri è quella di eseguire "troncature" a partire dai vertici dei poliedri regolari.

Per eseguire queste troncature basta considerare delle sfere di raggio variabile e centro nei vertici del poliedro: le loro intersezioni con gli spigoli del poliedro forniscono i vertici delle "facce di troncatura". Occorre una certa pazienza per ottenere i risultati voluti, ma non si tratta di tecniche complesse. Proponiamo, nella figura 13, un'immagine che mostra lo schema di questo processo nel caso in cui il poliedro di partenza sia un icosaedro. Si noti che i poligoni ottenuti dalla troncatura sono pentagoni regolari, mentre i poligoni "residui" delle facce originali dell'icosaedro sono esagoni non regolari.

Quando il raggio della sfera troncatrice raggiunge 1/3 della lunghezza degli spigoli dell'icosaedro, gli esagoni residui diventano regolari e si ottiene il cosiddetto *icosaedro troncato*: ha ovviamente 20 facce esagonali regolari (tante quante erano le facce dell'icosaedro originale) e 12 facce pentagonali regolari (tante quanti erano i vertici dell'icosaedro originale). Il risultato è proposto nella figura 14.

L'icosaedro troncato costituisce la forma base del pallone da calcio, il quale è la proiezione sulla sfera circoscritta dello stesso icosaedro troncato.

La rappresentazione grafica di un pallone da calcio richiede di poter costruire la proiezione, sulla sfera circoscritta, di tutti gli spigoli dell'icosaedro stesso, che sono 90; in realtà basta proiettare quelli visibili, perché chiaramente in una rappresentazione del pallone da calcio non ha molto interesse mostrare la struttura della "faccia nascosta".

La proiezione in questione è una proiezione centrale dal centro della sfera (che supponiamo essere l'origine per semplicità) sulla superficie della sfera

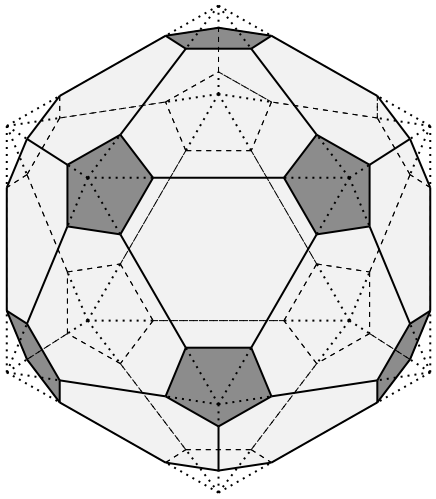

Figura 13: Schema della troncatura dell'icosaedro a partire dai vertici.

stessa. In linea di principio la cosa non è complessa e si articola, per ogni spigolo, nelle fasi di seguito descritte.

Supponiamo di avere un segmento $\overline{\mathsf{AB}}$ di estremi $(x_\mathsf{A}, y_\mathsf{A}, z_\mathsf{A})$ e $(x_\mathsf{B}, y_\mathsf{B}, z_\mathsf{B})$. Si comincia con il parametrizzare il segmento stesso, nel modo canonico:

$$\mathsf{P}(t)\colon \begin{cases} f(t) &= x_\mathsf{A} + (x_\mathsf{B} - x_\mathsf{A})t \\ g(t) &= y_\mathsf{A} + (y_\mathsf{B} - y_\mathsf{A})t \\ h(t) &= z_\mathsf{A} + (z_\mathsf{B} - z_\mathsf{A})t \end{cases}, \ 0 \le t \le 1.$$

Si calcola poi la norma di $P(t)$:

$$||\mathsf{P}(t)|| = \sqrt{f^2(t) + g^2(t) + h^2(t)}.$$

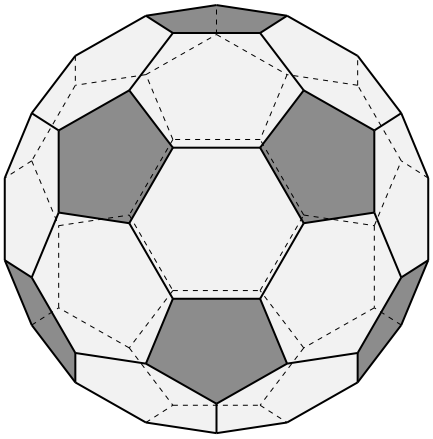

Figura 14: L'icosaedro troncato.

La proiezione del segmento $\overline{\mathsf{AB}}$ sulla sfera unitaria ha equazioni parametriche

$$\mathsf{Q}(t) = \frac{\mathsf{P}(t)}{||\mathsf{P}(t)||}.$$

A questo punto non resta altro da fare che usare la proiezione parallela già descritta per ottenere la stessa curva in 2D.

La figura 15 illustra il risultato di questo processo nel caso di un triangolo ABC.

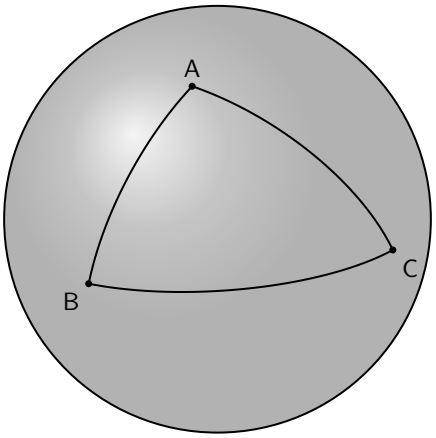

Figura 15: Proiezione centrale di un triangolo sulla sfera unitaria.

Rimane poi il problema del riempimento delle varie parti con il colore voluto, nel caso del pallone solo i pentagoni che sono colorati di nero.

Le formule della proiezione sono abbastanza semplici, tuttavia la loro scrittura in Geogebra richiede un bel po' di pazienza e la ricerca di strategie per velocizzare il procedimento: i codici per i vari segmenti differiscono infatti solo per i nomi dei punti, quindi una volta costruito un segmento basta fare un copia e incolla e modificare solo i nomi dei punti.

A puro titolo di curiosità riportiamo il codice Geogebra per il segmento $\overline{\mathsf{AB}}$ della figura 15: in ogni caso un copia e incolla di questo codice può essere utilizzato per qualunque segmento presente nella finestra 3D di Geogebra, con il solo cambio dei nomi degli estremi! Questo codice riproduce esattamente la combinazione delle formule di proiezione centrale e di proiezione parallela che abbiamo descritto.

```
Curva[((x(A)+(x(B)-x(A))t)/sqrt(((x(A)+
 (x(B)-x(A))t))^2+((y(A)+(y(B)-y(A))t))^2+
 ((z(A)+(z(B)-z(A))t))^2))x(i)+
 ((y(A)+(y(B)-y(A))t)/sqrt(((x(A)+
 (x(B)-x(A))t))^2+((y(A)+
 (y(B)-y(A))t))^2+((z(A)+
 (z(B)-z(A))t))^2))x(j)+
 ((z(A)+(z(B)-z(A))t)/sqrt(((x(A)+
 (x(B)-x(A))t))^2+((y(A)+
 (y(B)-y(A))t))^2+((z(A)+
 (z(B)-z(A))t))^2))x(k),((x(A)+
 (x(B)-x(A))t)/sqrt(((x(A)+
 (x(B)-x(A))t))^2+((y(A)+
 (y(B)-y(A))t))^2+((z(A)+
 (z(B)-z(A))t))^2))y(i)+
 ((y(A)+(y(B)-y(A))t)/sqrt(((x(A)+
 (x(B)-x(A))t))^2+((y(A)+
 (y(B)-y(A))t))^2+
 ((z(A)+(z(B)-z(A))t))^2))y(j)+
 ((z(A)+(z(B)-z(A))t)/sqrt(((x(A)+
 (x(B)-x(A))t))^2+((y(A)+
 (y(B)-y(A))t))^2+((z(A)+
 (z(B)-z(A))t))^2))y(k),t,0,1]
```

Il risultato finale nel caso in esame è proposto nella figura 16

Figura 16: Il pallone da calcio ottenuto mediante proiezione centrale su una sfera di un icosaedro troncato.

Un ultimo suggerimento pratico, prima di chiudere l'argomento "sportivo" di questo articolo. Il codice TikZ di una figura come quella del pallone

da calcio è abbastanza complesso (per la figura proposta è costituito da ben 250 linee) e non conviene costruire un unico maxi file in Geogebra, quanto piuttosto spezzarlo in piccoli file (per esempio uno per ogni esagono e una per ogni pentagono) e poi esportare il codice un pezzo alla volta. Questo sistema rende anche decisamente più agevole la colorazione delle varie parti.

8 Sezioni coniche e sferiche

La costruzione di grafici relativi alle sezioni coniche è uno dei cavalli di battaglia di chi si occupa di grafica geometrica 3D. Proponiamo qui alcuni esempi, senza particolari commenti: la tecnica base da utilizzare è quella ormai ampiamente descritta.

Cominciamo con il proporre una figura relativa ad una situazione poco nota e che è contenuta nella Proposizione I.5 del trattato di Apollonio[19] sulle coniche. La proposizione recita più o meno così: *Ci sono due serie di sezioni coniche circolari in un cono obliquo a base circolare, una serie costituita dalle sezioni parallele alla base, l'altra costituita dalle sezioni subcontrarie alla prima serie.*

La figura 17 illustra questo fatto. Si noti che abbiamo scelto di non visualizzare i paralleli della superficie conica, inoltre delle "linee meridiane" abbiamo mostrato solo quelle visibili: tutto questo per una migliore leggibilità della figura.

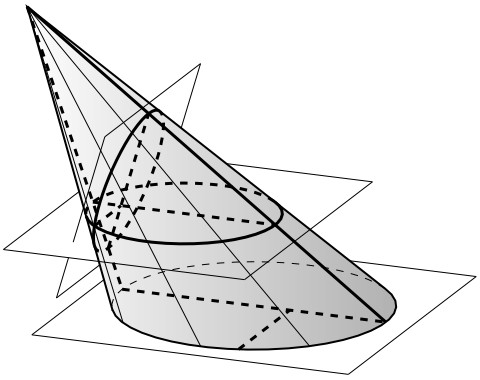

Figura 17: Le due serie di sezioni circolari in un cono obliquo a base circolare: sezioni parallele alla base e sezioni subcontrarie.

Non ci sono particolari difficoltà nella realizzazione in Geogebra 3D e successiva proiezione 2D, in quanto le linee coinvolte sono tutte coniche o segmenti. L'unico problema (di natura squisitamente matematica e non informatica) è quello di determinare l'inclinazione del piano di sezione per la sezione subcontraria: per fare questo basta seguire le indicazioni di Apollonio nella dimostrazione della già citata proposizione I.5.

19. Una versione in notazione moderna di questo celebre trattato in 8 libri, di cui solo i primi 4 ci sono pervenuti, si può trovare in Heath (1896).

La seconda immagine che proponiamo si riferisce ad una sezione conica vera e propria, precisamente l'iperbole. Tra le tante figure che si possono realizzare per questo problema abbiamo scelto la 18. Essa appare particolarmente complessa, per la separazione delle due parti del cono ottenute sezionandolo con un piano. In realtà in Geogebra basta realizzare un'unica figura del cono con l'iperbole sezione (realizzazione standard perché le linee coinvolte sono solo segmenti e coniche); successivamente si nasconde (con i comandi di Geogebra) prima una parte e poi l'altra, esportandole separatamente. I due frammenti di codice TikZ si uniscono spostando il secondo di alcuni centimetri: basta racchiudere il codice relativo come segue

```
\begin{scope}[xshift=2.4cm]
<codice della seconda parte>
\end{scope}
```

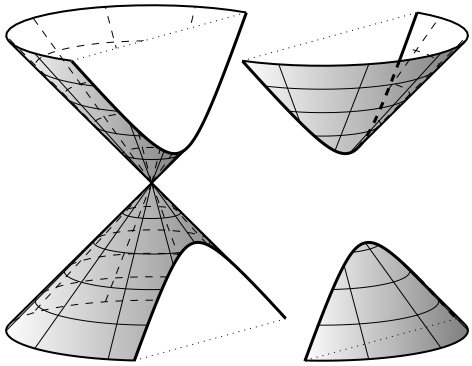

Figura 18: Sezione conica per ottenere un'iperbole.

La terza immagine che proponiamo in questa serie si riferisce all'intersezione tra un cilindro e una sfera e ci è stata suggerita durante le lezioni di un corso di Matematica per il design. Sculture galleggianti con questa forma sono state realizzate da Marta Pan[20] nel 1973 per il City Hall di Dallas.

La curva intersezione è una *finestra di Viviani* e il suo tracciamento è semplice solo se si utilizzano le relative equazioni parametriche, equazioni che comunque si possono trovare facilmente in letteratura. Il resto è ormai standard con i metodi indicati.

9 Un'ultima immagine

Chiudiamo questa trattazione sulla grafica 3D in TikZ con l'immagine di un poliedro, precisamente l'icosaedro elevato, suggeritaci da Claudio Beccari e che ha una lunga storia. Quasi certamente la sua prima pubblicazione si trova nel *De divina proportione* di Luca Pacioli, di cui sono conservate due copie (una a Ginevra e una a Milano) eseguite

20. Scultrice francese di origine ungherese, nata nel 1923 a Budapest e morta nel 2008 a Parigi.

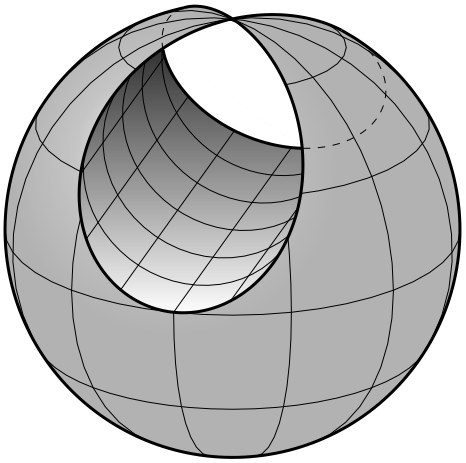

Figura 19: Intersezione tra un cilindro e una sfera: solido ottenuto asportando dalla sfera la parte di cilindro.

da diversi amanuensi verso la fine del 1400, oltre alle copie a stampa del 1509. I disegni sono redatti da Leonardo da Vinci, che rivela anche qui la sua mano maestra eseguendo i disegni con precisione fotografica. Ci sono due immagini di questo poliedro nel *De divina*, una del poliedro solido e una del poliedro vuoto: ne riproduciamo la prima (Icosaedron Elevatum Solidum), la seconda (Icosaedron Elevatum Vacuum) è decisamente troppo complessa (ma non è detto che non si possa fare!).

Il motivo per cui Claudio Beccari ci ha suggerito questo poliedro è il fatto che, a Urbino, è comunemente usato, nella sua forma leonardesca "vacua", per produrre lampadari in cui gli spigoli sono di ottone e le facce di vetro, con la lampadina al centro: il suo nome corrente è, appunto, *Stella di Urbino*.

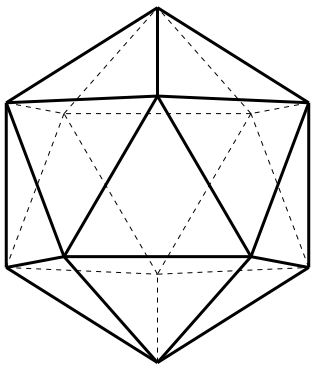

Figura 20: L'icosaedro.

Il solido deriva dall'icosaedro (il più complesso dei solidi platonici, con 20 facce costituite da triangoli equilateri), in cui ogni faccia è sostituita da un tetraedro regolare sporgente dall'icosaedro originale: ne proponiamo l'immagine nella figura 20.

Siccome ogni faccia viene sostituita da 3 facce (ancora triangoli equilateri) si ottiene un solido con 60 facce, tutte costituite da triangoli equilateri. Si tratta di un poliedro non convesso, anzi di un poliedro *stellato*. Esso non rientra comunque nella categoria dei poliedri stellati quasi-regolari di Keplero-Poinsot, ma è di estremo interesse avendo le facce tutte uguali (è una delle condizioni per cui un poliedro possa dirsi regolare: purtroppo mancano le altre).

La sua realizzazione in Geogebra è teoricamente standard in quanto si tratta di costruire, a partire da ogni faccia triangolare, un nuovo tetraedro che abbia quella faccia come base; in realtà è decisamente complessa in quanto occorre riuscire a gestire un elevato numero di facce, spigoli e vertici. Una volta eseguita la costruzione e la sua proiezione 2D, la cosa più delicata è valutare il migliore angolo visuale possibile, in modo da rendere efficacemente la struttura del solido. Qui non ha senso pensare di visualizzare in tratteggio o qualche altro modo gli spigoli nascosti: la figura diventerebbe illeggibile, tanto che nemmeno Leonardo ci ha provato!

È proprio nella ricerca del miglior angolo visuale possibile che si manifesta tutta la potenza del metodo grafico basato su un software di geometria dinamica: solo un genio come Leonardo ha potuto concepire la figura senza i moderni strumenti grafici (o fotografici). A parte questo fatto (molto delicato, lo ripetiamo), la costruzione non richiede tecniche particolari, in quanto si tratta di una figura costituita solo da segmenti. La figura 21 che proponiamo è sostanzialmente simile a quella di Leonardo, con un limitata variazione dell'angolo di inclinazione orizzontale.

Per una figura come questa, il colore sarebbe essenziale per una immediata comprensione della struttura completa del solido. Una strategia che può essere implementata, e che fornirebbe un sicuro aiuto nella facile lettura della figura, è quella dell'introduzione di un'opportuna ombreggiatura. In teoria non è una cosa complessa, soprattutto se si vuole una sorgente all'infinito e quindi con raggi paralleli: si tratterebbe di determinare, per ogni vertice, la sua ombra e, a partire da essa, costruire l'ombra di tutte le parti. Sicuramente la figura diventa complessa, ma sfruttando le capacità di Geogebra di visualizzare solo le parti di interesse, può essere implementata.

10 Conclusioni

Riteniamo che la tecnica proposta possa essere vantaggiosamente impiegata per la produzione di una gran parte delle figure tipo geometrico richieste in un testo di matematica. Accoppiata con pgfplots consente di produrre testi anche complessi utilizzando solo codice LaTeX, senza interventi di software esterni. Per produrre immagini molto complesse occorre naturalmente una buona conoscenza

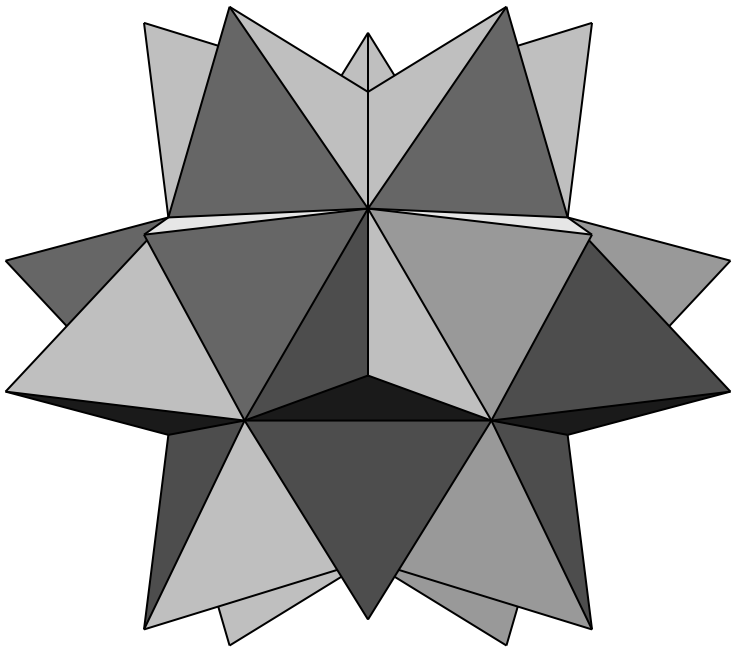

Figura 21: Icosaedro piramidato o "Stella di Urbino".

di Geogebra, ma secondo la nostra esperienza la sua curva di apprendimento è decisamente molto più piatta che non quella di TikZ. Inoltre l'uso di Geogebra presenta gli altri numerosi vantaggi che abbiamo descritto nell'articolo.

Naturalmente bisogna sempre tenere conto che non c'è rosa senza spina e non si può sperare di produrre effetti, come quelli che abbiamo mostrato, senza fatica e sperimentazione.

Chi è interessato a ulteriori approfondimenti può consultare il nostro sito http://www.batmath.it dove sono pubblicati numerosi materiali contenenti immagini costruite con il metodo qui presentato.

Riferimenti bibliografici

Barbaro, D. (1769). *La Pratica della Perspettiva*. Camillo & Rutilio Borgominieri fratelli, al Segno di S.Giorgio, Venetia.

Battaia, L. (2007). «LaTeX nella Scuola Media Superiore: applicazioni didattiche con PSTricks». *ArsTEXnica*, (4), pp. 45–50. URL http://www.guitex.org/home/numero-4.

Beccari, C. (2011). «The unknown picture environment». *ArsTEXnica*, (11), pp. 57–64. URL http://www.guitex.org/home/it/numero-11.

De Marco, A. (2007). «Illustrazioni tridimensionali con Sketch/LaTeX/PSTricks/TikZ nella didattica della Dinamica del Volo». *ArsTEXnica*, (4), pp. 51–68. URL http://www.guitex.org/home/numero-4.

— (2008). «Gestione avanzata delle figure in LaTeX». *ArsTEXnica*, (6), pp. 10–27. URL http://www.guitex.org/home/numero-6.

— (2009). «Produrre grafica vettoriale di alta qualità programmando asymptote». *ArsTEXnica*, (8), pp. 25–39. URL http://www.guitex.org/home/numero-8.

De Marco, A. e Giacomelli, R. (2011). «Creare grafici con pgfplots». *ArsTEXnica*, (12), pp. 12–38. URL http://www.guitex.org/home/it/numero-12.

Heath, T. L. (1896). *Apollonius of Perga - Treatise on Conic Sections edited in modern Notation*. Cambridge: at the University Press.

Voss, H. (2014). *3D plots: pst-3dplot*. URL https://www.ctan.org/pkg/pst-3dplot.

Wolcott, K. (2012). «Three-dimensional graphics with PGFTikZ». *TUGboat*, **33** (1), pp. 102–113.

▷ Luciano Battaia
Università Ca' Foscari
Dipartimento di Economia
luciano dot battaia at unive dot it